A Performance Evaluation of Detectors and Descriptors for UAV Visual Tracking

Bruce Cowan[1,3], Nursultan Imanberdiyev[1,2], Changhong Fu[1,2], Yiqun Dong[1,2], and Erdal Kayacan[1]
[1]School of Mechanical and Aerospace Engineering
[2]ST Engineering-NTU Corporate Laboratory
Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore
[3]Department of Electronic & Electrical Engineering, University of Strathclyde, Glasgow, G1 1XW, Scotland
E-mail: kfb12160@uni.strath.ac.uk, iman0005@e.ntu.edu.sg, {changhongfu, dongyq, erdal}@ntu.edu.sg

***Abstract*—This paper is made up of a series of performance evaluations of computer vision algorithms, namely detectors and descriptors. The OpenCV 3.1 implementations of these algorithms were used for these evaluations. The main purpose behind these evaluations was to determine the best algorithms to use for a UAV guidance system.**

I. Introduction

The project was to compare the performance of different computer vision algorithms with a view to implementing a tracking system for a UAV in the case of an emergency landing. This UAV would have a camera providing a stream of video to a computer. The landing target area would be selected by a user at the time of requirement, and this selected area would be tracked in subsequent frames of the video stream. In order for high quality tracking to be carried out, it is important to select the most appropriate detector and descriptor algorithms for visual tracking applications.

OpenCV 3.1 was used for this project, and Python was chosen to be used as the programming language, due to the author's familiarity with it, and the relative ease of manipulating data. Most of the code involves calls to functions from the native OpenCV library, so the use of Python does not come at a significant performance cost.

A. Detectors and Descriptors

Detectors are used to locate points of interest in an image, which are often corners or other regions of high contrast.

Descriptors are used to describe the locality of a keypoint in a compact manner. Tracking is achieved by comparing the descriptors in one image with descriptors in another image of the same scene. Matches are then found based on a *distance* measurement between these descriptors. The type of distance measurement used depends on the descriptor data type (floating point or binary).

B. Implementation of Detectors and Descriptors in OpenCV

There are a number of detector and descriptor algorithms available in OpenCV 3.1, which are shown in Table I. As this table shows, some algorithms implement detector functionality, some only descriptor functionality and others both.

Several algorithms, namely AKAZE, BRISK, KAZE, ORB, SIFT and SURF are implemented in OpenCV as a "unified" detector and descriptor. This allows the detection and description parts to be run with one function call.

Table I
Detector and descriptor algorithms available in OpenCV 3.1

Algorithm	Detector	Descriptor
AGAST	✓	
AKAZE	✓	✓
BRIEF		✓
BRISK	✓	✓
DAISY		✓
FAST	✓	
FREAK		✓
GFTT	✓	
KAZE	✓	✓
LATCH		✓
LUCID	✓	✓
MSER	✓	
ORB	✓	✓
SIFT	✓	✓
STAR	✓	
SURF	✓	✓

C. Hardware and Software Setup

For this project, a computer with an Intel Core i5-5200U running at 2.20 GHz using Fedora 23 was used. Since the Fedora repositories only include OpenCV 2.4, and also due to the fact that the patented algorithms (SIFT and SURF) were disabled in this version, it was necessary to manually compile OpenCV 3.1 from source.

The source code for the tests described in this paper is available at https://github.com/SuborbitalPigeon/fyp.

II. Related Work

Numerous performance evaluations of detector and descriptor algorithms have been published in the literature. This section discusses the methods used in these publications.

In [1], a comparison of several detector algorithms was undertaken. This paper also introduces the widely-used "Oxford" data set of test images, which is described in detail in Section II-A. It also introduces the concept of overlap error, which is defined in Section V-B. The paper also defines the concept of *repeatability*, used throughout the detector evaluation of Section IV.

In [2], an evaluation of descriptor methods is detailed. Descriptors were evaluated based on metrics known as *recall* and *precision*. The process followed in this paper closely matches that of this project, which is laid out in Section V. Similar to [1], this test makes use of the “Oxford” dataset.

[3] describes evaluation of both detectors and descriptors. Tracking for objects from a video stream was the topic of this paper, so the test data used was a collection of still image frames from a video. This paper also defines the version of repeatability used in Section IV of this paper. Also, *precision* is used to measure descriptor performance.

[4] is mainly concerned with the speed of feature detection and description. This forms the basis of the speed testing found in Section III of this paper. However, it reports the results for keypoint detection and descriptor extraction separately, as opposed to the total time taken for both to occur as in this paper.

Also discussed in the literature are several methods of tracking which make use of particle filters with machine learning, for example [5]. However, this paper concerns the performance of tracking using detector and descriptor algorithms.

A. Test Image Set

The images used for this project were downloaded,[1] these files were used for the tests discussed in [1], [2]. This data set is made up of 8 directories containing 6 image files each, with paired text files containing homography matrix coefficients, which can be used to convert the base image (*img1*) to each of the reference images (*img2* – *img6*).

Each set of images represent different transformations, including zoom changes, rotation and viewpoint changes.

III. Speed Testing

This section describes a method of testing the time taken by different combinations of detector and descriptor algorithms. These values are the total time taken to detect keypoints and to compute descriptors for a set of images. By testing the time taken for both detection and description algorithms to run, an idea is given of how much the time taken by the descriptor computation varies depending on the number of keypoints found by the detector.

Speed of detector and descriptor algorithms is an important factor in real time tracking. In order to be able to track an object in real time, keypoints have to be found, and descriptors computed for each frame of the video source. If these tasks take a long time to complete, the maximum tracking speed will be reduced. The absolute times taken are highly dependent on the environment on which the code is run. However, running this test on one specific machine will give a comparison between the relative speeds of the algorithms under test.

A. Speed Test Procedure

Every possible combination of detector and descriptor from OpenCV 3.1 were tested, apart from LUCID, which did not work correctly. All algorithms tested made use of their default parameters.

AKAZE and KAZE descriptors can only be computed using keypoints found with AKAZE or KAZE detectors, so this reduced the possible combinations from the simple product of all algorithms.

B. Results

Running the test over all the image sets yielded the results shown in Fig. 1, with one graph per detector, which indicate the amount of time (in ms) taken per image. It is expected that there is a variation in the time taken per image, due to the fact that the image sets are of different dimensions.

The average number of keypoints found per image is shown in Table II. Generally, different descriptors using the same detector algorithm have no effect on the number of keypoints found. However, some descriptors remove keypoints from the list of found keypoints from the detection phase, due to not being able to compute a descriptor for some of these keypoints.

From these results, several conclusions can be drawn:

- The KAZE, MSER, SIFT and SURF detectors are much slower than the others.
- The DAISY, LATCH, SIFT and SURF descriptors are relatively slow.
- High speed detectors include AGAST, BRISK, FAST, GFTT and ORB.
- High speed descriptors include BRISK, ORB and BRIEF.
- ORB, STAR, GFTT and MSER detectors find relatively few keypoints.
- AGAST and FAST detectors find relatively large numbers of keypoints.

C. Combined Speed Testing

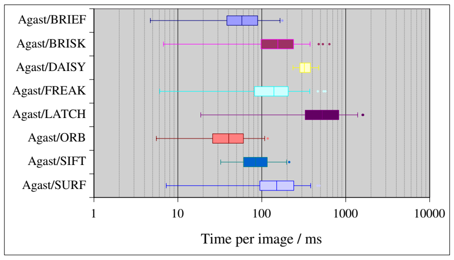
Several of the algorithms in OpenCV are implemented as “combined” detector and descriptor. This includes AKAZE, BRISK, KAZE, ORB, SIFT and SURF. They allow the use of a single function call to detect keypoints and compute descriptors.

1) Procedure: Each of the algorithms mentioned above is run on each file in the test image set. This only requires the creation of a single object for each algorithm, rather than separate ones for detector and descriptor. Due to the fact that different detectors find different numbers of keypoints for the same image, it is useful to report these results as time taken per keypoint. From this information, time per keypoint can be found.

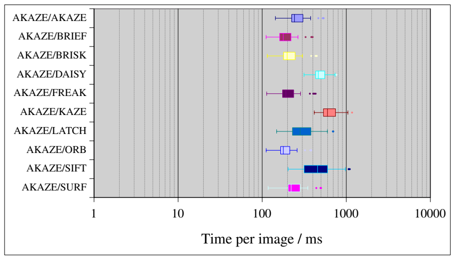
2) Results: Running over all the images in the test data set yielded the results shown in Table III. The “total time” column indicates the total amount of time taken to process all the images (in ms). A graph showing average time per keypoint is shown in Fig. 2. The values displayed are for each image in turn.

Comparing these results with those found in Section III-B, there is a slight advantage in speed by using a “combined” detector and and descriptor over separate detection and extraction stages. Several other notable features:

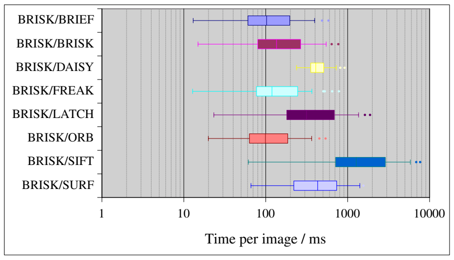
[1]From http://www.robots.ox.ac.uk/~vgg/research/affine/



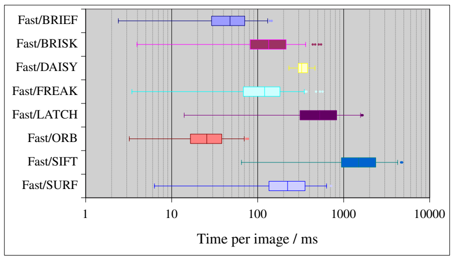
(a)



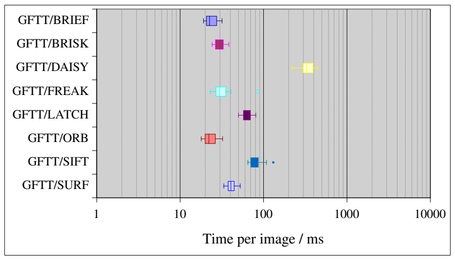
(b)



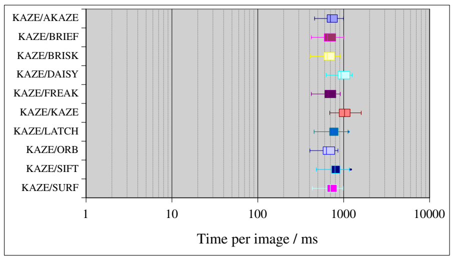
(c)



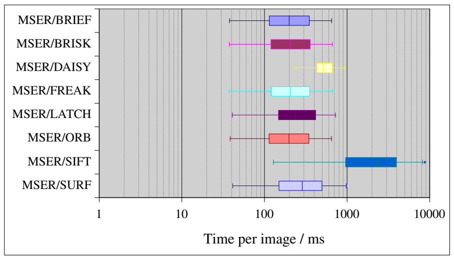
(d)



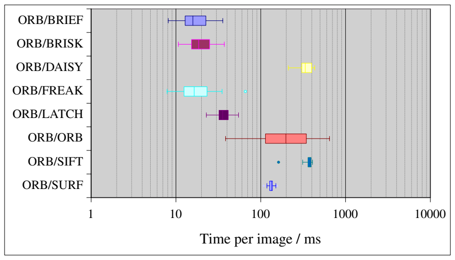
(e)



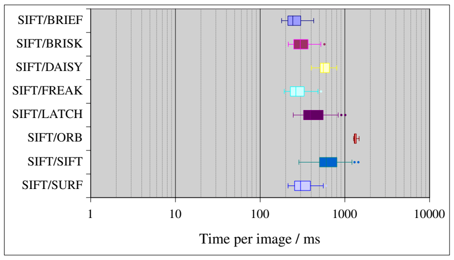
(f)



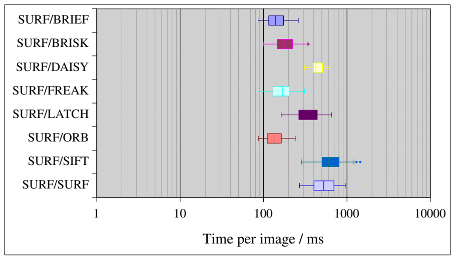
(g)



(h)



(i)



(j)

Figure 1. Speed Test Results

Table II
AVERAGE NUMBER OF KEYPOINTS FOUND IN THE SPEED TEST

	Detector										
Descriptor	Agast	AKAZE	BRISK	Fast	GFTT	KAZE	MSER	ORB	SIFT	SURF	Star
AKAZE		3087				2975					
BRISK	15 168	3087	8051	15 026	947	2796	932	417	5224	4729	915
KAZE		3087				2975					
ORB	13 818	3075	7803	13 717	874	2612	994	496		5002	915
BRIEF	14 034	3087	7873	13 928	885	2649	1011	496	4879	5070	915
DAISY	15 997	3087	8051	15 823	1000	2975	1220	496	5477	5363	915
FREAK	14 375	3087	6987	14 258	903	2635	804	245	4965	3755	908
LATCH	14 107	3087	7894	14 000	889	2662	1019	496	4903	5092	915
SIFT	15 997	3087	8051	15 823	1000	2975	1220	496	5477	5363	915
SURF	15 997	3087	8051	15 823	1000	2975	1220	496	5477	5363	915

Table III
RESULTS OF COMBINED SPEED TEST

Algorithm	Total time/ms	Total keypoints	Time per keypoint/µs
AKAZE	9137.58	148 192	80.0792
BRISK	7733.16	386 463	21.7529
KAZE	34 649.7	142 800	338.398
ORB	1000.2	23 830	42.085
SIFT	26 738.6	262 872	140.111
SURF	24 343.0	257 441	98.299

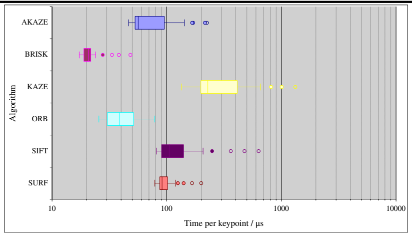


Figure 2. Combined speed test results

- KAZE and SIFT are relatively slow.
- SURF is about 30 % faster than SIFT.
- BRISK and ORB are the fastest algorithms.
- ORB finds far fewer keypoints than the others (by default, it is set to find 500 per image).

IV. REPEATABILITY

Repeatability is an often used [1], [6] measure of how *stable* the keypoints returned by a detector are. An ideally stable keypoint would be one which does not change location when the image undergoes a geometric transformation with respect to the transformation. This is a also a measure of a detector's invariance.

Repeatability is calculated by detecting keypoints in the base image (*img1*) and each of the reference images (*img2* – *img6*). The set of keypoints found in the base image is designated S_b and the set for the reference image S_r. The keypoints from the reference image are reprojected according to the inverse of the homography associated with the image pair (which is defined as the homography from the base to reference images). As reprojecting an image with rotation, translation or zoom changes causes a void area to be created, keypoints which are outside the common area of these images are removed from both sets. By determining the Euclidean distance between every keypoint in the base image ($\boldsymbol{x}_b$) with those in the reprojected reference image ($\boldsymbol{x}_r$), the number of repeating keypoints can be found. A keypoint is repeating if there is a keypoint in the reprojected image within 2 pixels of a keypoint in the base image. This metric is known as ϵ-repeatability [6], with ϵ in this case set to 2 pixels.

Repeatability is then calculated by using the following equation [6], [7]:

$$\text{repeatability} = \frac{\left|\left\{(\boldsymbol{x}_b \in S_b, \boldsymbol{x}_r \in S_r) \;\middle|\; \left\|\boldsymbol{x}_b - \boldsymbol{H}^{-1}\boldsymbol{x}_r\right\| < \epsilon\right\}\right|}{|S_b|} \tag{1}$$

where $\boldsymbol{H}$ is a the homography matrix.

A. Results

Graphs showing repeatability for the test sets are shown in Fig. 3. These graphs show for increasing transformation magnitude (represented by larger image numbers), the repeatability decreases. The graphs also indicate the number of keypoints found in the base image before transformation. Detectors which find a large number of keypoints are more likely to have higher repeatability, because there is no way of knowing whether a repeating keypoint pair are actually the "same" keypoint, or just a coincidentally close pair of keypoints.

From these results, it can be seen that the repeatability of detector algorithms has a strong dependency on the type of transformation included in the test image set. This would suggest that a detector should be selected based on the expected image transforms. In the case of a UAV, these would include translation, rotation and scale changes. Therefore, the most important sets would be *bark* and *boat*. Due to the camera on the UAV pointing downwards, perspective changes are less important, the corresponding sets are *graf* and *wall*. AGAST, BRISK or ORB may yield the best results as far as choosing a detector for this purpose goes.

V. PRECISION AND RECALL

Precison and recall are commonly used to characterise the quality of a matching process [8]. These are used to determine the quality of descriptors, and how well they can be used to match keypoints between two images. Given two images, the distances between all the points' descriptors in image 1 and image 2 are found. A matching pair of descriptors is the pair of descriptors from image 1 and image 2 where the distance between them is minimised.

Recall and precision are defined as follows:

$$\text{recall} = \frac{\text{number of correct positives}}{\text{total number of positives}} \tag{2}$$

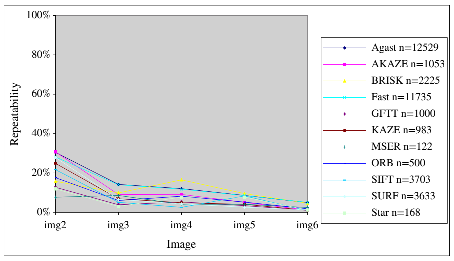
$$1 - \text{precision} = \frac{\text{number of false positives}}{\text{total number of matches}} \tag{3}$$

By varying the threshold between what are regarded as correct and false positives, different values of recall and 1-precision can be calculated. These values can be plotted on a graph. An ideal descriptor would have a recall of 100 % no matter what the precision is, i.e. it would return all the correct matches, but no incorrect ones.

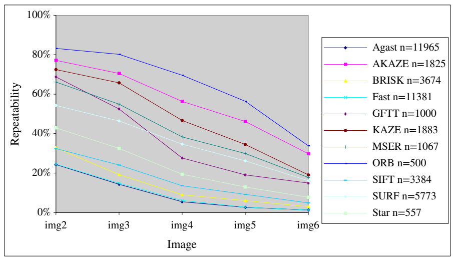
A. Test method

It was decided to use nearest neighbour thresholding for this test. It was easier to implement the changing threshold required to produce the graphs for this test using this method.

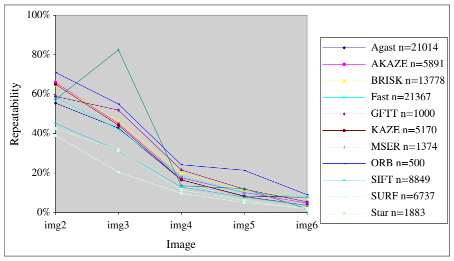
Two images are used for the precision-recall calculation, *img1* and another image from the same set. SURF features are found for both images, and these are stored. SURF was used due to its relative stability under geometric transformation. This use of SURF means that AKAZE and KAZE descriptors could not be tested. Attempting to use AKAZE as the detector yielded poor results.



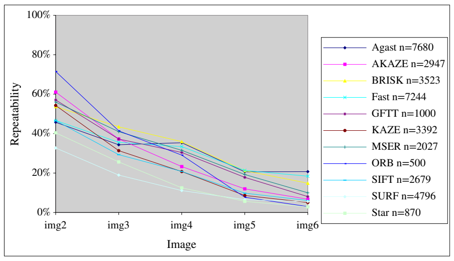
(a)



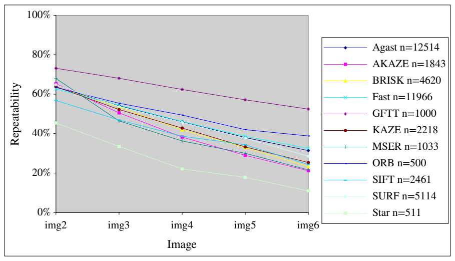
(b)



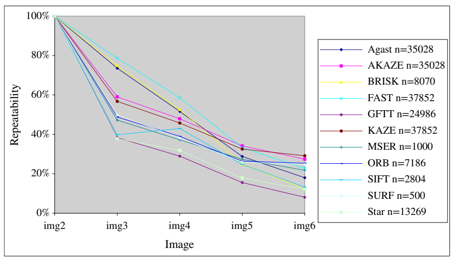
(c)



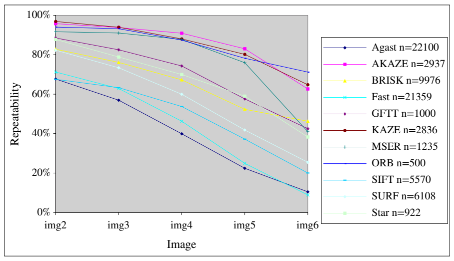
(d)



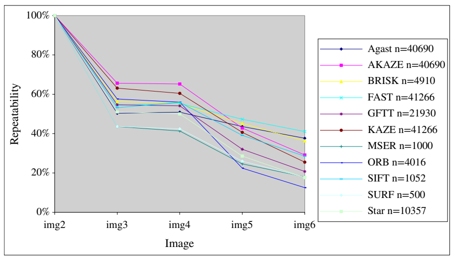
(e)



(f)



(g)



(h)

Figure 3. Repeatability Test Results

Iterating through all the descriptor algorithms, the following steps were carried out: Descriptors were extracted for each image, and a brute force matcher was used to match the descriptors. Matches were removed where either of the corresponding keypoints were outside the common image area. A list of corresponding keypoints between the two images was found. This made use of the overlap error between the keypoints. Pairs of points with an overlap error of $\epsilon < 0.4$ were regarded as corresponding points.

B. Overlap Error

In order to determine if two points correspond, a criterion must be defined. OpenCV represents keypoints as circular areas with a certain radius. For this test, overlap error was used. The overlap error between two regions A and B is defined by.

$$\text{overlap error} = 1 - \frac{A \cap B}{A \cup B} \tag{4}$$

1) Thresholding: Once the matches and corresponding matches are known, precision and recall can be found using (3) and (2). However, in order to generate graphs, the threshold between a *correct* and *false* match has to be varied. Therefore the minimum and maximum distances for the matches were found, and a threshold variable was set up to vary between these values. Iterating through these values of threshold, and finding the matches whose distance was below this threshold, the number of *correct* and *false* matches are determined.

C. Results

By running the test on a file from each of the test image sets, the results of Fig. 4 were generated. Depending on the complexity of the transformation applied to each set, the magnitude was varied by using a different reference image number. This was required to make sure enough corresponding matches existed for all of the algorithms under test.

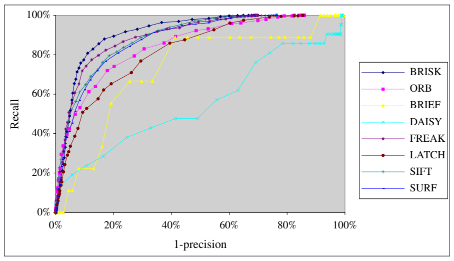
From these results, it can be seen that DAISY and BRIEF had significant problems with this procedure. It is possible that the choice of detector for this test (SURF) had an effect on these descriptors more than any of the others.

The other descriptor algorithms performed similarly most of the time. However, generally BRISK outperformed the other algorithms at high precision. The tests where this wasn't the case (*graf* and *wall*) both exhibit large perspective changes.

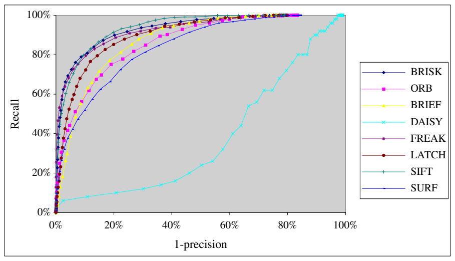
Due to the nature of the camera set up of the UAV to be used, the best descriptors are BRISK, FREAK or ORB.

VI. Conclusion

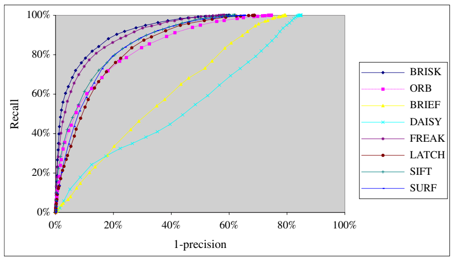
Object tracking makes use of matching pairs of descriptors in one video frame and subsequent frames. Time taken for feature detection and description is of primary importance



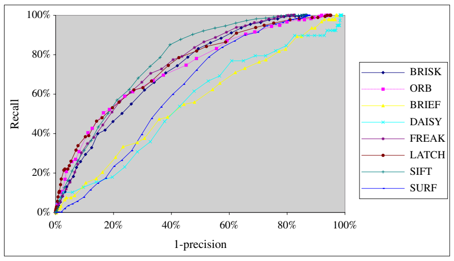
(a)



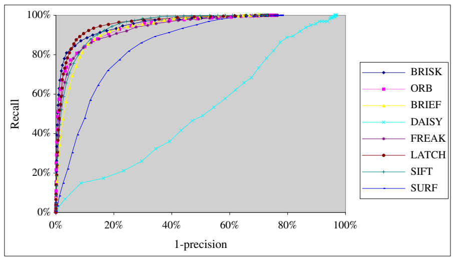
(b)



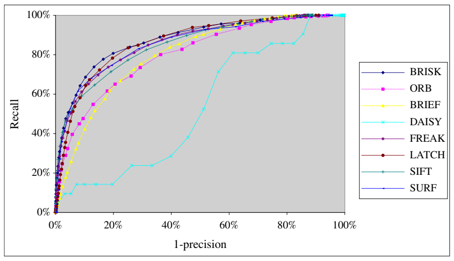
(c)



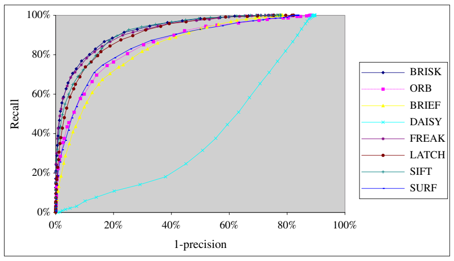
(d)



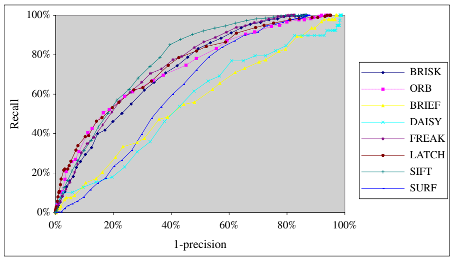
(e)



(f)



(g)



(h)

Figure 4. Precision-recall Test Results

in order to maximise the tracking frequency. The chosen detector's repeatability also affects the tracking performance. Since tracking involves matching descriptors, the descriptor's performance also has an effect.

Acknowledgements

I would like to thank my supervisor, Dr. Kayacan, as well as Dr. Fu and Mr. Imanberdiyev for their assistance and advice during this project. I also thank the people of Singapore for being so warm and welcoming during my stay in the country.

The research was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme.

References

[1] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. V. Gool, 'A comparison of affine region detectors', *Int. J. Comput. Vision*, vol. 65, no. 1, pp. 43–72, Nov. 2005.

[2] K. Mikolajczyk and C. Schmid, 'A performance evaluation of local descriptors', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[3] S. Gauglitz, T. Höllerer and M. Turk, 'Evaluation of interest point detectors and feature descriptors for visual tracking', *Int. J. Comput. Vision*, vol. 94, no. 3, pp. 335–360, Sep. 2011.

[4] O. Miksik and K. Mikolajczyk, 'Evaluation of local detectors and descriptors for fast feature matching', in *2012 21st Int. Conf. Pattern Recognition*, pp. 2681–2684.

[5] S. Zhang, H. Zhou, H. Yao, Y. Zhang, K. Wang and J. Zhang, 'Adaptive normalhedge for robust visual tracking', *Signal Process.*, vol. 110, pp. 132–142, May 2015.

[6] C. Schmid, R. Mohr and C. Bauckhage, 'Evaluation of interest point detectors', *Int. J. Comput. Vision*, vol. 37, no. 2, Jun. 2000.

[7] S. Ehsan, N. Kanwal, A. F. Clark and K. McDonald-Maier, 'Improved repeatability measures for evaluating the performance of feature detectors', *Electron. Lett.*, vol. 46, pp. 998–1000, 14 Jul. 2010.

[8] Y. Ke and R. Sukthankar, 'Pca-sift: A more distinctive representation for local image descriptors', in *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, vol. 2, pp. 506–513.